Homework 1

I data Analyst dell'Associazione Nazionale Musei Italiani sono interessati ad analizzare il ricavo medio per biglietto. In particolare, vorrebbero che le analisi affrontassero i seguenti aspetti.
Un museo ha un nome univoco e si trova in una città specifica. Vengono memorizzate anche la provincia e la regione in cui il museo risiede. La stessa città può ospitare diversi musei. Ogni museo appartiene ad una categoria specifica (ad esempio, "Arte", "Siti storici", "Storia naturale").
Un museo può avere alcuni servizi aggiuntivi disponibili per il suo pubblico. I sistemi registrano quali servizi sono disponibili per ogni museo. Esempi di servizi aggiuntivi sono "visite guidate", "audio guide", "guardaroba", "caffè", "Wi-Fi". Il numero di servizi aggiuntivi è 10 e la loro lista completa è nota.
I biglietti venduti da ogni museo sono registrati. Ci sono 3 diversi tipi di biglietti: "Full Revenue", "Reduced-student" (per studenti dai 14 ai 24 anni) e "Reduced-junior" (per giovani con meno di 14 anni).
I sistemi memorizzano anche come viene acquistato il biglietto. Un biglietto può essere acquistato in tre modalità: online, nelle biglietterie autorizzate, o direttamente all'ingresso del museo.
Le analisi devono essere effettuate considerando la data, il mese, il bimestre, il trimestre, il semestre, l'anno, se la data è un giorno lavorativo o festivo, e la fascia oraria di validità del biglietto. La fascia oraria è memorizzata in 3 intervalli di blocchi di 4 ore (08:00-12:00, 12:01-16:00, 16:01- 20:00).

L'azienda è interessata alle statistiche sul ricavo medio per biglietto. L'analisi deve essere effettuata sulla base di:

- nome del museo, categoria del museo, città, provincia, regione
- servizi del museo
- tipo di biglietto (intero, ridotto-studenti, ridotto-junior)
- modalità di acquisto (online, nelle biglietterie autorizzate o all'ingresso del museo)
- data di validità del biglietto, mese, bimestre, trimestre, semestre, giorno lavorativo e fascia oraria

Tasks

1. Progettare il data warehouse per rispondere alle specifiche e per rispondere in modo efficiente a tutte le query fornite. Disegnare lo schema concettuale del data warehouse e lo schema logico (tabelle dei fatti e delle dimensioni).

BIGLIETTO (IdBiglietto, IdMuseo, IdTempo, Modalita`, Tipo, Entrate, Numero, Fascia)
MUSEO (IdMuseo, Nome, Citta`, Provincia, Regione, Categoria, (... Servizi ...))
TEMPO (IdTempo, Data, Mese, Bimestre, Trimestre, Semestre, Anno, Lav)

2. Scrivere le seguenti query usando il linguaggio SQL esteso:
   a. Separatamente per ogni tipo di biglietto e per ogni mese (della validità del biglietto), analizzare: le entrate medie giornaliere, le entrate cumulative dall'inizio dell'anno, la percentuale di biglietti relativi al tipo di biglietto considerato sul numero totale di biglietti del mese
   b. Considerare i biglietti del 2021. Separatamente per ogni museo e tipo di biglietto analizzare: il ricavo medio per un biglietto, la percentuale di ricavo sul ricavo totale per la categoria di museo e tipo di biglietto corrispondenti, assegnare un rango al museo, per ogni tipo di biglietto, secondo il numero totale di biglietti in ordine decrescente.

a
```
SELECT TIPO, MESE, ANNO, sum(ENTRATE)/COUNT(DISTINCT Data)
       sum(sum(ENTRATE) OVER( PARTITION BY TIPO
                              ORDER BY ANNO ROWS UNBOUNDED PRECEDING)
       sum(sum(ENTRATE) OVER( PARTITION BY MESE
                              ORDER BY ANNO ROWS UNBOUNDED PRECEDING)
       100 x sum(sum(NUMERO) / sum(sum(NUMERO)) OVER(PARTITION BY MESE)
FROM   BIGLIETTI B, TEMPO T
WHERE  B.IdTempo = T.IdTempo
GROUP BY TIPO, MESE, ANNO
```

b
```
SELECT   NOME, TIPO, sum(ENTRATE)/Sum(NUMERO)
         100 x SUM(ENTRATE)/ sum(sum(ENTRATE)) OVER (PARTITION BY TIPO, CATEGORIA)
         RANK OVER(PARTITION BY TIPO
                   ORDER BY sum(NUMERO)
FROM   BIGLIETTO B, TEMPO T, MUSEO M
WHERE  T.IdTempo = B.IdTempo,  M.IdMuseo = B.IdMuseo
       AND T.ANNO = '2021'
GROUP BY NOME, TIPO, Categoria
```

3. Creare e mantenere aggiornate una vista materializzata con i comandi CREATE MATERIALIZED VIEW e CREATE MATERIALIZED VIEW LOG di ORACLE

   Considerare le seguenti query di interesse:

   - Analizzare le entrate medie mensili relative ad ogni tipo di biglietto e per ogni semestre.
   - Separatamente per ogni tipo di biglietto e per ogni mese analizzare le entrate cumulative dall'inizio dell'anno.
   - Considerando solo i biglietti acquistati online, separatamente per ogni tipo di biglietto e per ogni mese analizzare il numero totale di biglietti, le entrate totali e le entrate medie per biglietto.
   - Separatamente per ogni tipo di biglietto e per ogni mese analizzare il numero totale di biglietti, le entrate totali e le entrate medie per biglietto per l'anno 2021.
   - Analizzare la percentuale di biglietti relativi ad ogni tipo di biglietto e mese sul numero totale di biglietti del mese.

   ```
   CREATE MATERIALIZED VIEW VM1
   BUILD IMMEDIATE
   REFRESH FAST ON COMMIT
   AS
     SELECT Tipo, Mese, Anno, SUM(Numero) AS SNum, SUM(Entrate) AS SEnt, Semestre, Modalità
     FROM BIGLIETTI B, TEMPO T
     WHERE B.IdTempo = T.IdTempo
     GROUP BY Tipo, Mese, Anno, Semestre, Modalità
   ```

   ```
   CREATE MATERIALIZED VIEW LOG ON BIGLIETTI
   WITH SEQUENCE, ROWID (Tipo, Numero, Modalità, IdTempo, Prezzo)
   INCLUDING NEW VALUES

   CREATE MATERIALIZED VIEW LOG ON TEMPO
   WITH SEQUENCE, ROWID (IdTempo, Mese, Anno, Semestre)
   INCLUDING NEW VALUES
   ```

   Le operazioni sono UPDATE, DELETE, INSERT

4. Aggiornamento e gestione delle viste tramite Trigger
   Supponendo che il comando CREATE MATERIALIZED VIEW non sia disponibile, creare la vista materializzata definita nell'esercizio precedente e definire la procedura di aggiornamento a partire da modifiche sulla tabella dei fatti realizzata tramite trigger.

   a. Creare la struttura della vista materializzata con CREATE TABLE VM1 (...)

b. Popolare opportunamente la tabella creata con il seguente comando
   INSERT INTO VM1 (...)
   (SELECT ...
   ...)

c. Scrivere il trigger necessario per propagare le modifiche (inserimento di un nuovo record) effettuate nella tabella dei FATTI alla vista materializzata VM1.

d. Specificare quali operazioni (ad esempio INSERT) attivano il trigger creato al punto c

```
CREATE TABLE VM1(
  Tipo int,
  Mese int,
  Anno int,
  SNum float,
  SEnt float,
  Semestre int,
  Modalita int
);

INSERT INTO VM1
(SELECT Tipo, Mese, Anno, SUM(Numero) AS SNum, SUM(Entrate) AS SEnt, Semestre, Modalità
  FROM BIGLIETTI B, TEMPO T
  WHERE B.IdTempo = T.IdTempo
  GROUP BY Tipo, Mese, Anno, Semestre, Modalità
  );

CREATE OR REPLACE TRIGGER VM1
AFTER INSERT ON BIGLIETTO
FOR EACH ROW
DECLARE
  N NUMBER
  VarMese, VarAnno, VarTipo, VarSemestre, VarModalità int
BEGIN
  ---LETTURA DEI VALORI DEGLI ATTRIBUTI NECESSARI PER L'AGGIORNAMENTO
  SELECT Mese, Semestre, Anno
  INTO VarMese, VarSemestre, VarAnno
  FROM TEMPO
  WHERE IdTempo = :NEW.IdTempo;

  SELECT Tipo, Modalità
  INTO VarTipo, VarModalità
  FROM BIGLIETTO
  WHERE IdBiglietto = :NEW.IdBiglietto;

  ---VERIFICO SE NELLA VISTA BIGLIETTO (FATTO) ESISTE UNA TUPLA CON I VALORI LETTI PRIMA
  SELECT COUNT(*) INTO N
  FROM VM1
  WHERE Mese = VarMese
  AND Tipo = VarTipo
  AND Anno = VarAnno
  AND Semestre = VarSemestre
  AND Modalità = VarModalità;
```

```
  if N>0 then
    ---SE LA TUPLA E' GIA' PRESENTE ALLORA AGGIORNO
    UPDATE VM1
      SET SEnt = Ent + :NEW.Entrate
          SNum  = Num + :NEW.Numero
      WHERE Mese = VarMese
      AND Tipo = VarTipo;
    else
    ---SE LA TUPLA NON E' ANCORA PRESENTE ALLORA INSERISCO
    INSERT INTO BIGLIETTO(Tipo, Mese, Anno, SNum, SEnt, Semestre, Modalità)
    VALUES (VarTipo, VarMese, VarAnno, :NEW.SNum, :NEW.SEnt, VarSemestre, VarModalità);
  endif
end
```

Ad attivare il trigger sarà l’operazione INSERT INTO BIGLIETTO